

数据库

MySQL是一个数据库管理系统（软件，DBMS），用来进行数据管理的；

- 数据库：用来存放数据的；
- 表：是真正存储数据的地方；
- 列：字段、属性；
- 记录：一条数据就是一个记录；
- 数据：列和记录交叉的地方；

变量类型

1. 数字类型：

- int
- smallint
- tinyint
- bigint
- float
- double
- decimal

2. 文本类型：字符串类型

a) Char：指定的固定长度

- 长度是固定的：不管存储多少数据，在磁盘里面占用的长度是确定；
- 性能要比varchar好一些；

b) Varchar：可变的char

- 最大空间是固定的，实际存储空间根据实际的数据长度来定；
- 性能会受损；

c) Text：用来存非常大的文本信息，比如新闻信息、产品描述信息等；

d) Longtext：

3. 时间类型：

a) 日期：2022-08-01

b) 时间：13:57

c) Datetime：2022-08-01 13:57:24

表的设计

三个范式：通俗理解三个范式是我们设计表的基本指导思想

- 范式1：保证字段的原子性（不可再分了）和唯一性（字段之间不能重复）；
- 范式2：在满足范式1的情况之下，保证每条记录只干一件事情：主键，id sid；
- 范式3：在满足范式2的条件之下，不传递依赖；外键：我们主观上知道是外键就可以了，不要在表

里面关联，防止一些不必要的错误发生。

表的设计思想：一事情一张表，一张表做一事情

用户信息一张表；

文章信息一张；

点赞：谁（点赞的用户）在什么时间（添加时间）点赞了 哪篇文章（文章的ID）

SQL 语句语法规则

MySQL 数据库的操作分类

根据数据库的对象层级，可以将SQL的基础操作分为四类：

- 数据库（DB）操作
- 数据表（Table）操作
- 数据字段（Field）操作
- 数据操作

一、数据库（DB）的基本操作

在终端的任何位置，输入如下命令，即可进入 mysql 命令的执行窗口：

```
mysql -u root -p
```

1、创建数据库

语法格式：

```
1 create database 数据库名称 [数据库选项];
```

数据库名称的命名规范：

由数字、字母和下划线组成。

不区分大小写。

不能以数字开头。

建议使用下划线法创建复杂的数据库名字。比如

my_db_01。

举例：

创建一个名为 my_db_01 的数据库：

```
1 create database my_db_01;
```

创建一个指定字符集的数据库：

```
1 create database my_db_02 charset utf8MB4;
```

创建一个指定校对集的数据库：

```
1 create database my_db_03 charset utf8MB4 collate utf8mb4_general_ci
```

2、查看数据库

查看有哪些数据库：（显示所有的数据库列表）

```
1 show databases;
```

查看 my_db_01 这个数据库的具体创建指令是怎样的：

```
1 show create database my_db_01;
```

备注：由于系统会加工，所以看到的结果不一定是真实的创建指令。

3、使用指定的数据库

使用指定的数据库：（也可以理解成：进入指定的数据库）

语法格式：

```
1 use 数据库名称;
```

举例：

```
1 use my_db_01;
```

4、修改数据库的参数

修改字符集

修改校对集

语法格式：

```
1 alter database 数据库名称 [库选项];
```

举例1、修改数据库的字符集为gbk：

```
1 alter database my_db_01 charset gbk;
```

举例2、修改数据库的校对集：

```
1 alter database my_db_01 charset gbk collate gbk_chinese_ci;
```

备注：因为校对集是和字符集有关的，所以上方指令是在修改字符集的同时，修改校对集。

5、删除指定的数据库

语法格式：

```
1 drop database 数据库名称;
```

备注：删除数据库时，会清空当前数据库里的所有数据表，所以删除数据库的操作一定要谨慎。

二、数据表（Table）的基本操作

注意，我们最好先通过 use xxx_database 命令进入指定的数据库（DB），然后在当前数据库下，进行数据表（Table）的操作。

1、创建数据表

语法格式：

```
1 create table [数据库名].[表名] ( 字段名1 字段类型, ...字段名2 字段类型) 表选项;
```

举例：

1、在当前数据库中创建数据表 users，并新增主键 id 字段：

```
1 CREATE TABLE users (uid int NOT NULL AUTO_INCREMENT PRIMARY KEY);
```

2、在当前数据库中创建数据表 t_stu，并新增 name、age 这连个字段：

```
1 create table t_stu (name varchar(255), age int);
```

3、在指定的数据库 my_db_02 中创建数据表 t_student2：

```
1 create table my_db_02( name varchar(255), age int);
```

4、在当前数据库中创建数据表 t_student3（含表选项）：

```
1 create table t_student3( name varchar(255), age int)engine InnoDB charset utf8MB4;
```

5、显示数据表的名称

在当前数据库下，显示所有的数据表：

```
1 show tables;
```

在指定的数据库中，显示所有的数据表：

```
1 show tables from my_db_01;
```

显示数据表的创建指令：(查看 users 这个数据表的具体创建指令是怎样的)

```
1 show create table users ; # 备注：由于系统会加工，所以看到的结果不一定是真实的创建指令。
```

6、desc：查看数据表的表结构

查看数据表的表结构，就是查看这张表中定义了哪些字段，以及这些字段是如何定义的。通过这种方式，我们可以清晰地了解数据的存储形式。

项目开发中，领导在检查我们的工作时，首先看的就是我们的表中定义了哪些字段。所以说，这种方式，还是很实用的。

语法格式：

```
1 方式1: desc 表名称;  
2 方式2: describe 表名称;  
3 方式3: show columns from 表名称;
```

7、修改数据表的表名称和表选项

修改数据表的表名称：

在当前数据库下，修改数据表的表名称：

```
1 rename table 原表名 to 新表名;
```

指定某个数据库，然后修改数据表的表名称：

```
1 rename table 数据库名.原表名 to 数据库名.新表名;
```

修改数据表的表名称：

```
1 alter table table1 charset gbk;
```

8、删除数据表

语法格式：

```
1 drop table 数据表名称;
```

三、数据的基本操作

1、新增数据

方式1、全字段插入：

语法格式：

```
1 insert into 表名 values(值1, 值2, ... 最后一个值);
```

解释：

值的顺序必须与所有字段的顺序一致。

值的数据类型也必须与字段定义的数据类型一致。

举例（给 t_qiangu1 这个表中插入一条完整的数据）：

```
1 insert into users(3, '12345698','123456','http://baidu.com/1.png','靓仔',);
```

方式2、部分字段插入：

语法格式：

```
1 insert into 表名 (字段1, 字段2, 字段3) values(值1, 值2, 值3);
```

解释：

字段的顺序可以随意，但值的顺序必须要与前面的字段顺序一一对应，数据类型也要一致。

举例（给 users 这个表中的指定字段插入数据）：

```
1 INSERT INTO users (tel,passwd,avator,nickname) VALUES  
('1236495','12346','http://baidu.com/2.jpg','健身');
```

2、修改数据

语法格式：

```
1 update 表名 set (字段1 = 新值1, 字段2 = 新值2) [where 条件筛选];
```

解释：

通常结合 where 条件语句来修改数据。

3、删除数据

删除字段的操作不可逆，请谨慎操作。

语法格式：

```
1 delete from 表名 [where 条件];
```

解释：

执行删除操作之后，匹配到的整条记录，都会删除。删除数据之前，要先保证表里面有数据。如果这张表是空表，那么，执行这个命令后，等于没执行。

举例：

删除表中id = 2的记录：

```
1 delete from users where id = 2;
```

4、查询数据

查询数据的操作，占sql日常操作的95%以上。

语法格式：

```
1 select xxx from 表名;
```

举例：

查询表中的所有数据：

```
1 select * from users;
```

查询表中 tel、nivkname 这两个字段的数据：

```
1 select tel,nivkname from users;
```

根据where条件查询，查询表中 id=2 的数据：

```
1 select * from users where id = 2;
```

指定范围查询：

```
1 select * from yg where salary between 6000 and 8000;
```

指定具体的值，不再是范围，

```
1 select * from yg where id in(8,10,18,38,48);
```

模糊查询

```
1 select * from yg where name like '%小%';  
2 select * from yg where name like '王%';  
3 select * from yg where name like '%锋';
```

MySQL 5.7 Command Line Client

```
mysql> select * from yg where name like '%小%';  
+----+-----+-----+-----+  
| id | name   | salary | deptid |  
+----+-----+-----+-----+  
| 22 | 姚小妮 | 8088   | 2      |  
| 24 | 苟小晶 | 7896   | 4      |  
| 37 | 王小溥 | 8096   | 2      |  
+----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

```
mysql> select * from yg where name like '王%';  
+----+-----+-----+-----+  
| id | name   | salary | deptid |  
+----+-----+-----+-----+  
| 8  | 王游   | 6018   | 3      |  
| 32 | 王丁锋 | 7065   | 2      |  
| 37 | 王小溥 | 8096   | 2      |  
+----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

```
mysql> select * from yg where name like '%锋';  
+----+-----+-----+-----+  
| id | name   | salary | deptid |  
+----+-----+-----+-----+  
| 32 | 王丁锋 | 7065   | 2      |  
+----+-----+-----+-----+  
1 row in set (0.00 sec)
```

排序

```
1 select * from yg where salary >7000 order by deptid asc, salary desc;
```

asc: 默认为升序

desc: 降序

分页:

```
1 SELECT
2     查询列表
3 FROM
4     表1 别名1
5     【连接类型】 JOIN 表2 别名2 ON 连接条件
6     【WHERE 分组前的筛选】
7     【GROUP BY 分组字段】
8     【HAVING 分组后的筛选】
9     【ORDER BY 排序字段 ASC|DESC】
10 LIMIT 【offset, 】size ;
11
12 select * from pc where 1 order by id asc limit 0,10;
```

- limit (start, maxlength):
- start: 开始位置, 如果是0, 可以省略;
- maxlength: 最大长度

特点:

- limit语句放在查询语句的最后
- offset代表起始索引, 起始索引从0开始, size代表条目个数
- 分页语句: select 查询列表 from 表 limit (page-1)*size,size;

```
1 每页显示: size = 50 page=13;
2 Page1: SELECT * FROM pc WHERE 1 ORDER BY id ASC LIMIT 0, 50;
3 Page2: SELECT * FROM pc WHERE 1 ORDER BY id ASC LIMIT 50, 50;
4 Page3: SELECT * FROM pc WHERE 1 ORDER BY id ASC LIMIT 100, 50;
5
6 pagen: SELECT * FROM pc WHERE 1 ORDER BY id ASC LIMIT (page - 1) *size, size;
```

合计函数 MAX、MIN、AVG、SUM、COUNT、NOW

```
1 select avg(salary) as avgsal, max(salary) msal, min(salary) minsal, sum(salary) total
   from yg;
```

avg: 求平均值

max: 最大值

min: 最小值

sum: 求和

```

1 select count(*) as nums from yg where salary >8000;
2 SELECT MAX(salary) FROM yg;
3 SELECT MIN(salary) FROM yg;

```

分组查询

```

1 SELECT
2     查询列表
3 FROM
4     表
5     【where 筛选条件】
6 GROUP BY 分组的字段
7     【having 分组后的筛选】
8     【order BY 排序的字段】 ;
9
10 1、和分组函数一同查询的字段必须是group by后出现的字段
11
12 2、筛选分为两类：分组前筛选和分组后筛选
13
14         针对的表          语句位置          连接的关键字
15 分组前筛选    分组前的原始表    group by前    where
16 分组后筛选    分组后的结果集    group by后    having
17
18
19 3、分组可以按单个字段也可以按多个字段
20
21 4、分组可以搭配着排序使用
22
23 select avg(salary) avgсал, deptid from yg group by deptid;

```

HAVING查询：WHERE不支持合计函数

```

1 select
2     avg(salary) avgсал,
3     deptid
4 from
5     yg
6 group by
7     deptid
8 having avg(salary) > 8000;

```

多表查询

```
1 select
2     a.aid,
3     a.title,
4     u.avator,
5     u.nickname
6 from
7     articles a,
8     users u
9 where
10     a.aid = 3 and a.uid = u.uid;
```

JOIN的方式实现多表连接查询

内连接: inner

外连接:

- 左外连接: left (左边的是主表)
- 右外连接: right (右边的是主表)
- 全外连接: full (两边都是主表, 但是MySQL不支持全外连接、Oracle支持)

交叉连接: cross (交叉连接其实是用sql99语法实现笛卡尔乘积)

```
1 1 left:
2 SELECT
3     yg.*,
4     d.dname
5 from
6     yg
7 left join
8     dept d
9 on
10     yg.deptid = d.deptid
11 where
12     yg.salary > 8000;
13
14
15 2 right:
16 select
17     yg.*,
```

```

18     d.dname
19 from
20     yg
21 right join
22     dept d
23 on
24     yg.deptid =d.deptid
25 where
26     yg.salary>8000;
27
28
29 3 inner:
30 select
31     yg.*,d.dname
32 from
33     yg
34 inner join
35     dept d
36 on
37     yg.deptid = d.deptid
38 where
39     yg.salary >8000;

```

子查询

- 子查询放在小括号内
- 子查询一般放在条件的右侧
- 子查询的执行优先于主查询执行，主查询的条件用到了子查询的结果
- 标量子查询，一般搭配着单行操作符使用：>、>=、<、<=、!=、<>、=、<=>
- 列子查询，一般搭配着多行操作符使用：in、not in、any、some、all、exists

要求：查询 每个部门里面 工资 高于 部门平均工资 的员工；

```

1 第一步：
2 select avg(salary) avgsal, deptid from yg group by deptid;
3 第二步：把一步的代码作为第二步的一个表进行查询，给表as个别名
4 select
5     yg.*,
6     avgt.avgsal,
7     d.dname

```

```

8  from
9      yg, dept d,(select avg(salary) avgsal, deptid from yg group by deptid) as avgt
10 where
11     yg.salary > avgt.avgsal and yg.deptid = avgt.deptid and yg.deptid = d.deptid;

```

按照部门分组：

```

1  SELECT
2      COUNT(*) nums,deptid
3  from
4      (SELECT
5          yg.* ,
6          avgt.avgsal,
7          d.dname
8      FROM
9          yg, dept d, (SELECT AVG(salary) avgsal,deptid FROM yg GROUP BY deptid)AS avgt
10     WHERE
11         yg.salary > avgt.avgsal AND yg.deptid = avgt.deptid AND yg.deptid = d.deptid)AS tab
12     GROUP BY tab.deptid;

```

where和having后面：

标量子查询：查询最低工资的员工姓名和工资

```

1  select
2      yg.name,
3      salary
4  from
5      yg
6  where salary=(
7      select min(salary) from yg
8  );

```

案例：

1.查询student表的所有记录 5分

```
1 SELECT
2 *
3 FROM
4 student;
5
```

student (11r × 6c)						
sid		sname	sex	birth	department	address
1		andy	女	2000	计算机系	四川省成华区
2		red	男	1999	数学系	贵州省遵义市
3		小五	女	2001	中文系	湖南省长沙市
4		李四	男	2002	英语系	贵州省贵阳市
5		王二	女	2000	中文系	四川省青阳区
6		tom	男	2002	计算机系	湖南省衡阳市
7		李四	女	2001	英语系	贵州省贵阳市
8		王浩	男	2000	中文系	四川省金牛区
9		李四	女	2001	数学系	四川省成都市
10		陈里	女	1999	计算机系	贵州省兴义市
11		顾客	男	2000	数学系	湖南省湘潭市

2.查询出学生第二到第四条记录5分

```
1 SELECT
2 *
3 FROM
4 student
5 LIMIT
6 1,3;
7
```

student (3r × 6c)						
sid		sname	sex	birth	department	address
2		red	男	1999	数学系	贵州省遵义市
3		小五	女	2001	中文系	湖南省长沙市
4		李四	男	2002	英语系	贵州省贵阳市

3.随机查出10个学生的信息 5分

```

1 SELECT
2   *
3 FROM
4   student
5 ORDER BY
6   RAND()
7 LIMIT
8 10;
9
10

```

student (10r × 6c)

sid	sname	sex	birth	department	address
6	tom	男	2002	计算机系	湖南省衡阳市
8	王浩	男	2000	中文系	四川省金牛区
7	李四	女	2001	英语系	贵州省贵阳市
4	李四	男	2002	英语系	贵州省贵阳市
10	陈里	女	1999	计算机系	贵州省兴义市
1	andy	女	2000	计算机系	四川省成华区
9	李四	女	2001	数学系	四川省成都市
2	red	男	1999	数学系	贵州省遵义市
11	顾客	男	2000	数学系	湖南省湘潭市
3	小五	女	2001	中文系	湖南省长沙市

4.查出数学系和汉语言文学系的学生 5分

```

1 SELECT
2   *
3 FROM
4   student
5 WHERE
6   department='数学系'
7 or
8   department='中文系';
9
10

```

student (6r × 6c)

sid	sname	sex	birth	department	address
2	red	男	1999	数学系	贵州省遵义市
3	小五	女	2001	中文系	湖南省长沙市
5	王二	女	2000	中文系	四川省青阳区
8	王浩	男	2000	中文系	四川省金牛区
9	李四	女	2001	数学系	四川省成都市
11	顾客	男	2000	数学系	湖南省湘潭市

5.查出20到25岁的学生 5分使用now()获取当前时间

```

1 SELECT
2 *
3 FROM
4 student AS s
5 WHERE
6 NOW()-s.birth>20
7 or
8 NOW()-s.birth<25;
9
10

```

student (11r × 6c)					
sid	sname	sex	birth	department	address
1	andy	女	2000	计算机系	四川省成华区
2	red	男	1999	数学系	贵州省遵义市
3	小五	女	2001	中文系	湖南省长沙市
4	李四	男	2002	英语系	贵州省贵阳市
5	王二	女	2000	中文系	四川省青阳区
6	tom	男	2002	计算机系	湖南省衡阳市
7	李四	女	2001	英语系	贵州省贵阳市
8	王浩	男	2000	中文系	四川省金牛区
9	李四	女	2001	数学系	四川省成都市
10	陈里	女	1999	计算机系	贵州省兴义市
11	顾客	男	2000	数学系	湖南省湘潭市

6.查出20到25岁的学生的成绩信息，要求显示学生名，课程名和分数 5分

```

1 SELECT
2 s.sname,c.c_name,c.grade
3 FROM
4 student s, score c,(SELECT * FROM student AS s WHERE NOW()-s.birth>20 or NOW()-s.birth<25)AS sage
5 WHERE
6 s.sid = c.stu_id
7 AND
8 s.sname=sage.sname;
9
10

```

student (18r × 3c)		
sname	c_name	grade
王二	英语	74
tom	计算机	90
tom	英语	55
李四	计算机	95
李四	英语	89
王浩	计算机	75
王浩	中文	86
陈里	英语	82
陈里	数学	84
andy	数学	65
小五	数学	85
tom	数学	90
李四	数学	78

7.查询每个院系各有多少人 5分


```

1 SELECT
2     COUNT(*), department
3 FROM
4     student
5 GROUP BY
6     department;
7
8
9

```

student (4r × 2c)	
COUNT(*)	department
3	中文系
2	数学系
1	英语系
3	计算机系

8.查出没有成绩的学生 5分

```

1 SELECT
2     *
3 FROM
4     student s, score c
5 WHERE
6     c.grade
7 NOT IN
8 (SELECT s.*, c.grade FROM student s, score c WHERE s.sid=c.stu_id)
9 AND
10    s.sid=c.stu_id;

```

9.查出每个学生的总成绩5分

```

mysql> select sname, SUM(grade) from score
-> LEFT JOIN student ON score.stu_id = student.sid GROUP BY sname;
+-----+-----+
| sname | SUM(grade) |
+-----+-----+
| andy  | 243        |
| red   | 271        |
| 小五  | 140        |
| 李8   | 244        |

```

10.查出小五的考试科目和考试成绩5分

```

1 SELECT
2   *
3 FROM
4   score as s
5 INNER JOIN
6   student as st
7 ON
8   s.stu_id = st.sid
9 WHERE
10  sname
11  LIKE
12  '小五';
13
14

```

结果 #1 (2r × 10c)

id	stu_id	c_name	grade	sid	sname	sex	birth	department	address
5	3	中文	55	3	小五	女	2001	中文系	湖南省永州市
25	3	数学	85	3	小五	女	2001	中文系	湖南省永州市

11.查出姓王的考试科目和考试成绩5分

```

1 SELECT
2   *
3 FROM
4   score as s
5 INNER JOIN
6   student as st
7 ON
8   s.stu_id = st.sid
9 WHERE
10  sname
11  LIKE
12  '王%';
13
14

```

结果 #1 (9r × 10c)

id	stu_id	c_name	grade	sid	sname	sex	birth	department	address
8	5	英语	74	5	王二	女	2000	汉语言文学系	福建省厦门市
9	6	计算机	90	6	王六	男	2002	计算机系	湖南省衡阳市
10	6	英语	55	6	王六	男	2002	计算机系	湖南省衡阳市
13	8	计算机	75	8	王浩	男	2000	汉语言文学系	辽宁省阜新市
14	8	中文	86	8	王浩	男	2000	汉语言文学系	辽宁省阜新市
28	5	数学	84	5	王二	女	2000	汉语言文学系	福建省厦门市
29	6	数学	90	6	王六	男	2002	计算机系	湖南省衡阳市
30	6	数学	85	6	王六	男	2002	计算机系	湖南省衡阳市
32	8	数学	80	8	王浩	男	2000	汉语言文学系	辽宁省阜新市

12.查询计算机成绩低于95的学生信息 5分

```

1  SELECT
2      sname
3  FROM
4      score
5  LEFT JOIN
6      student
7  ON
8      score.stu_id = student.sid
9  GROUP BY
10     c_name
11  HAVING
12     AVG(grade)<95;
13
14
15

```

student (4r × 1c)

sname

red

李8

andy

andy

13.从score表中查询每个科目的最高分 5分

```
mysql> select c_name, max(grade) from score group by c_name;
```

c_name	max(grade)
中文	90
数学	92
英语	95
计算机	98

4 rows in set (0.00 sec)

14.用连接的方式查询所有学生的信息和考试信息 5分

```
mysql> select * from score s left join student st on s.stu_id = st.sid;
```

id	stu_id	c_name	grade	sid	sname	sex	birth	department	address
1	1	计算机	98	1	andy	女	2000	计算机系	北京市海淀区
2	1	英语	80	1	andy	女	2000	计算机系	北京市海淀区
3	2	计算机	95	2	red	男	1999	数学系	北京市昌平区
4	2	中文	88	2	red	男	1999	数学系	北京市昌平区
5	3	中文	55	3	小五	女	2001	中文系	湖南省永州市
6	4	计算机	70	4	李四	男	2002	英语系	辽宁省阜新市
7	4	英语	92	4	李四	男	2002	英语系	辽宁省阜新市
8	5	英语	74	5	王二	女	2000	汉语言文学系	福建省厦门市
9	6	计算机	90	6	王六	男	2002	计算机系	湖南省衡阳市
10	6	英语	55	6	王六	男	2002	计算机系	湖南省衡阳市
11	7	计算机	95	7	李四	女	2001	英语系	湖南省衡阳市
12	7	英语	89	7	李四	女	2001	英语系	贵州省贵阳市
13	8	计算机	75	8	王浩	男	2000	汉语言文学系	辽宁省阜新市
14	8	中文	86	8	王浩	男	2000	汉语言文学系	辽宁省阜新市
15	9	中文	90	9	李四	女	2001	计算机系	四川省成都市
16	9	计算机	80	9	李四	女	2001	计算机系	四川省成都市
17	10	英语	82	10	李8	男	2003	英语系	湖南省长沙市
18	10	数学	84	10	李8	男	2003	英语系	湖南省长沙市
19	11	计算机	81	11	陈里	女	1999	计算机系	辽宁省阜新市
20	11	英语	95	11	陈里	女	1999	计算机系	辽宁省阜新市
21	12	计算机	98	12	顾客	男	2000	数学系	湖南省湘潭市
22	12	英语	80	12	顾客	男	2000	数学系	湖南省湘潭市
23	1	数学	65	1	andy	女	2000	计算机系	北京市海淀区
24	2	数学	88	2	red	男	1999	数学系	北京市昌平区
25	3	数学	85	3	小五	女	2001	中文系	湖南省永州市
26	4	数学	70	4	李四	女	2002	英语系	辽宁省阜新市
27	4	数学	92	4	李四	男	2002	英语系	辽宁省阜新市
28	5	数学	84	5	王二	女	2000	汉语言文学系	福建省厦门市
29	6	数学	90	6	王六	男	2002	计算机系	湖南省衡阳市
30	6	数学	85	6	王六	男	2002	计算机系	湖南省衡阳市
31	7	数学	78	7	李四	女	2001	英语系	贵州省贵阳市
32	8	数学	80	8	王浩	女	2000	汉语言文学系	辽宁省阜新市
33	9	数学	65	9	李四	女	2001	计算机系	四川省成都市
34	10	数学	78	10	李8	女	2003	英语系	湖南省长沙市
35	11	数学	75	11	陈里	女	1999	计算机系	辽宁省阜新市
36	12	数学	80	12	顾客	男	2000	数学系	湖南省湘潭市

36 rows in set (0.01 sec)

15.计算每个学生的总成绩 6分

```

1  SELECT
2      sname,SUM(grade)
3  FROM
4      score
5  LEFT JOIN
6      student
7  ON
8      score.stu_id = student.sid
9  GROUP BY
10     sname;
11
12
13
14

```

student (10r × 2c)	
sname	SUM(grade)
andy	243
red	271
小五	140
李8	244
李四	821
王二	158
王六	320
王浩	241
陈里	251
顾客	258

16.计算每个考试科目的平均成绩 6分

```

mysql> select c_name,AVG(grade) from score GROUP BY c_name;
+-----+-----+
| c_name | AVG(grade) |
+-----+-----+
| 中文   | 79.7500    |
| 数学   | 79.9333    |
| 英语   | 80.8750    |
| 计算机 | 86.8889    |
+-----+-----+
4 rows in set (0.00 sec)

```

17.查询同时参加计算机和英语考试的学生的信息 6分

18.将计算机考试成绩按从高到低进行排序 6分


```
mysql> select * from score where c_name='计算机' ORDER BY grade DESC;
```

id	stu_id	c_name	grade
1	1	计算机	98
21	12	计算机	98
3	2	计算机	95
11	7	计算机	95
9	6	计算机	90
19	11	计算机	81
16	9	计算机	80
13	8	计算机	75
6	4	计算机	70

9 rows in set (0.00 sec)

19.查询都是湖南的学生的姓名、年龄、院系和考试科目及成绩6分

```

1 SELECT
2   *
3 FROM
4   score as s
5 INNER JOIN
6   student as st
7 ON
8   s.stu_id= st.sid
9 WHERE
10  address
11  LIKE
12  '湖南%';
13
14

```

结果 #1 (12r × 10c)

id	stu_id	c_name	grade	sid	sname	sex	birth	department	address
5	3	中文	55	3	小五	女	2001	中文系	湖南省永州市
9	6	计算机	90	6	王六	男	2002	计算机系	湖南省衡阳市
10	6	英语	55	6	王六	男	2002	计算机系	湖南省衡阳市
17	10	英语	82	10	李8	男	2003	英语系	湖南省长沙市
18	10	数学	84	10	李8	男	2003	英语系	湖南省长沙市
21	12	计算机	98	12	顾客	男	2000	数学系	湖南省湘潭市
22	12	英语	80	12	顾客	男	2000	数学系	湖南省湘潭市
25	3	数学	85	3	小五	女	2001	中文系	湖南省永州市
29	6	数学	90	6	王六	男	2002	计算机系	湖南省衡阳市
30	6	数学	85	6	王六	男	2002	计算机系	湖南省衡阳市

